ELSEVIER International Journal of Approximate Reasoning 18 (1998) 251–270

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

# A parallel algorithm for building possibilistic causal networks

Ramón Sangüesa *, Ulises Cortés [1], Antonio Gisolfi [2]

*Department of Software, Technical University of Catalonia, Jordi Girona, 1-3, Barcelona 08034, Spain*

Received 1 March 1997; accepted 1 October 1997

**Abstract**

Among the several representations of uncertainty, possibility theory allows also for the management of imprecision coming from data. Domain models with inherent uncertainty and imprecision can be represented by means of possibilistic causal networks that, the possibilistic counterpart of Bayesian belief networks. Only recently the definition of possibilistic network has been clearly stated and the corresponding inference algorithms developed. However, and in contrast to the corresponding developments in Bayesian networks, learning methods for possibilistic networks are still few. We present here a new approach that hybridizes two of the most used approaches in uncertain network learning: those methods based on conditional dependency information and those based on information quality measures. The resulting algorithm, POSSCAUSE, admits easily a parallel formulation. In the present paper POSSCAUSE is presented and its main features discussed together with the underlying new concepts used. 

* Corresponding author. E-mail: sanguesa@lsi.upc.es.

[1] E-mail: ia@lsi.upc.es. This work has been supported by project CICYT-TIC960878 of the Spanish Science and Technology Commission.

[2] E-mail: gisolfi@dia.unisa.it. The research leading to this paper has been supported by the EHRC project VIM: A Virtual Multicomputer.

0888-613X/98/$19.00 
PII: S0888-613X(98)00010-3

## 1. Introduction

An important problem in Artificial Intelligence is that of modeling the knowledge of a given domain. It is usually the case that the domain to be modeled is an *ill-structured domain*. These kinds of domains are those whose structures are not well known, that is, those domains where the fundamental concepts are ill-defined (i.e. are ambiguous, imprecise or uncertain) and the relationships that exist between bodies of knowledge (represented as classes or variables) are also not very well defined or are inherently difficult to define. This kind of domains pose specially difficult problems for knowledge acquisition and learning. Some learning methodologies try to cope with such unstructuredness by means of multiple descriptions of concepts or some other kind of classification-based schemas [3]. However, when acquiring knowledge from a domain, relationships are as important as the basic concepts. Relationships may be of many types, not only the hierarchical ones implied by classification. *Causal associations* are specially useful in order to reveal the structure of a domain. Moreover, knowing what causes a given situation to appear, allows to detect which information is really relevant to take a decision and then solve the corresponding problem. In a certain sense, causal information sorts out relevant information and guides inference through the most appropriate knowledge available. This is specially true and useful in diagnosis and prediction tasks.

Extracting causal information from a domain may be done by recovering the structure of causal relationships between the bodies of knowledge in the domain. That is, of relationships that obey some criteria for causal association [25]. The problem of ill-structured domains is the inherent uncertainty about the existence of a true relationship between bodies of knowledge. When the model of the domain is described in terms of variables and their causal associations, the uncertainty appears at least at two levels:

1. uncertainty about the values that a variable can take,
2. uncertainty about which variables are causally related to a given one and with which strength.

The first type of uncertainty can be modeled by means of probability distributions on the values of the variables; the second one by means of conditional probability distributions establishing which is the probability that a given variable, say $x$, takes a certain value conditioned on the fact that a second variable, let us call it $y$, has taken another given value. Notice that in such a case if two variables are conditionally dependent, then one of them can be said to be the cause of the other (there exist several criteria relating conditional association and causality, see [9]). When such a modelization scheme is selected, the natural representation for the resulting model is a Bayesian Belief Network, which is a special type of Directed Acyclic Graph (DAG). We will define it in the following section. However, we are interested in the case when variables' values

are inherently *imprecise*. This is the case, for example, when data about the domain come from sensor readings which are certain *up to* a certain precision. In this case, possibility theory is a good alternative for representing such *imprecision under uncertainty*. The task is then to recover a causal structure from data exhibiting such properties. The resulting model is a possibilistic causal network, another type of DAG.

In the following sections we will describe Directed Acyclic Networks as a representation of conditional dependence information, and we will introduce our formalization of possibilistic networks. In Section 3 we will describe how possibilistic conditional dependence between variables is measured and how this information can be used to recover the underlying DAG in the data, in this section a new measure of possibilistic conditional dependence is introduced; in Section 4 the POSSCAUSE parallel algorithm is discussed; in Section 6 experimental results are commented. Finally, Section 7 summarizes our present work and outlines future developments.

## 2. DAG representations of conditional information

Given a domain of variables $U$ it is assumed that some dependence, specially conditional dependence, relationships exist among the variables in the domain.

**Definition 2.1** (*Dependence Model*). A list of assertions reflecting the existing dependencies is called a Dependence Model of the domain [29]. Assertions in the model are of the form $I(x|y|z)$ that are to be interpreted as '$x$ is independent of $z$ given $y$'. If two variables $x$ and $y$ are marginally independent then the assertion $I(x|\emptyset|y)$ is true.

Now, a Generalized Belief Network is a DAG that represents the conditional dependence relationships among variables in the domain.

**Definition 2.2** (*Generalized Belief Network*). For a domain $U = \{x_1 \ldots x_n\}$ the corresponding Generalized Belief Network is a DAG where nodes stand for variables and links for direct association between variables. Each link is quantified by the conditional uncertainty distribution relating the variables connected to it, $\mathscr{P}$. By uncertainty distribution we mean a distribution resulting from quantifying uncertainty by means of a confidence measure.

Belief networks have two interesting characteristics. Firstly, any given node $x_i$ in a belief network is conditionally independent of the rest of the variables in $U$, given its direct predecessors in the graph, i.e., its parents 'shield' the variable from the influence of the previous variables in the graph. Secondly, the joint uncertainty distribution induced by the DAG representing the dependences

in a given domain can be factored out into the conditional distribution of each variable with respect to its immediate predecessors ('parents'). That is

$$\mathscr{P}(x_1 \ldots x_n) = \otimes \mathscr{P}(x_i|pa_i),$$

where $\mathscr{P}$ represents an uncertainty distribution (probability, possibility, etc.) and $\otimes$ is a factorizing operator. In the case of probability this is the product of conditional distributions.

**Definition 2.3** (*Possibilistic Causal Networks*). Possibilistic causal networks are belief networks whose underlying uncertainty distribution is the possibility distribution defined on the variables of the domain.

## 3. Possibilistic conditional dependency measures

In possibility theory, there exists several conditioning operators and, moreover, dependence measures can be defined in several ways (see [8] for a discussion). A natural way of defining conditional independence is in terms of information irrelevance [10,11]. That is, $x$ is conditionally independent from $y$ given $z$ ($I(x|z|y)$) if knowing a new information about $z$ induces no changes in the existing relationship between $x$ and $y$. How are these changes defined and measured? We defined a measure that is guided by a very simple rationale: given two variables $x$ and $y$, the similarity between the possibility distributions $\pi(x)$ and $\pi(x|y)$ may be used as an indication about the mutual relevance of the two variables. If $I(x|\emptyset|y)$, then $\pi(x)$ and $\pi(x|y)$ are identical because no change is induced on $x$ values by knowing some new information on $y$. The more different $\pi(x)$ and $\pi(x|y)$ are, the more dependent $x$ and $y$ are. The same argument can be extended for comparing the resulting effect when a third variable, $z$, is used for conditionalization. A simple way to measure such a change is then the comparison of changes in the *form* of the corresponding distributions. In order to do so, several points are to be taken into account.

1. Only changes in possibility of a certain degree are significant, given the natural variability of distributions coming from data.
2. Changes at several points may be more important than in a single point depending of the possibility of occurrence of the corresponding values.
3. Changes in more possible values should be more important than changes in less possible values.

What is envisaged is a measure of relevance that, taking into account the changes in form, should be also able to measure the amount of possibility mass lost or won after conditioning. The greater the changes are the greater the dependence is.

Now, in measuring dependences from data one cannot be too strict, room for *imprecision in similarity* has to be provided. In this sense, we admit a *graded*

similarity between possibility distributions. Given two possibility distributions $\pi_{x_i}$, $\pi_{y_i}$ the set of points that are *dissimilar* to a certain degree $\alpha$ is the $\alpha$-set.

**Definition 3.1** ($\alpha$-*set*). Given two variables $x_i$, and $y_i$, taking values in their respective domains $X$ and $Y$; given a real value $\alpha$ the $\alpha$-set is the set of points $x_i$ in $X$ such that $|\pi_X(x_i) - \pi_{X|Y}(x_i)| \leqslant \alpha$

$$\alpha\text{-set} = \{x_i \in X | \pi_X(x_i) - \pi_{X|Y}(x_i)| \geqslant \alpha\}.$$

Note that the cardinality and the precise elements of the $\alpha$-set depend on the value of $\alpha$. When $\alpha$ is zero, only those $x_i$ whose values for $\pi_X(x_i)$ and $\pi_{X|Y}(x_i)$ are identical will become part of the $\alpha$-set. When $\alpha$ is equal to one, then all points belong to the set.

Once we have defined the set of *informative points*, i.e., those that belong to the $\alpha$-set, it is easy to define a measure of similarity between possibility distributions. The rationale is to take into account only those points that belong to the $\alpha$-set and measure the difference in possibility that conditioning on values of a second variable $Y$ induces in the original distribution. Given a variable $X$ taking values $x_1 \ldots x_n$ what is measured is how much each $\pi_X(x_i)$ differs from $\pi_{X|Y}(x_i)$. Now, each difference in value is weighted by the possibility of occurrence of the corresponding $y_i$ value, $\pi_Y(y_i)$.

By assessing the similarity between the corresponding distributions, one can compare the influence of different variables on $X$ and rank them in order of conditional dependence.

**Definition 3.2** (*Conditional Dependence Degree between two variables*). Given two variables $X$ and $Y$ with respective possibility distribution $\pi_x$ and $\pi_y$, and a real value $\alpha$, the dependence degree between $X$ and $Y$, $Dep(X, Y, \alpha)$, is

$$\mathrm{Dep}(x, y, \alpha) = \sum_{y_i \in Y} \pi(y_i) \sum_{x_i \in \alpha\text{-set}} |\pi(x_i) - \pi' y(x_i|y_i)|$$

With this definition of strength of conditional association of two variables one can extract from data a list of conditional dependency assertions which may be the base for constructing the corresponding DAG. Methods for recovering possibilistic DAGs from data are discussed in the following section.

## 4. Causal network construction

The problem of belief network construction can be cast at least in two different forms. Firstly, a belief network can be recovered by using dependency information extracted from the data or from a given expert. Starting from a dependency assertion list a DAG that is a perfect map ([22,29]) for such a list

has to be recovered. This is the base for *Dependence Based Recovery Methods* [2,5,17,18,7,23,24,30]. A second family of methods is based on information quality measures [15,6,13,21]. In this case the idea is to recover a DAG whose underlying joint probability minimizes distance from the assumed joint distribution in data or that minimizes a given information quality measure. A measure of the first type of criterion is Kulback–Leibler cross-entropy [20]. A measure of the second type of criterion is entropy modified as to reflect the overall joint entropy of a DAG in terms of the factorization due to the belief network structure as Cooper and Herskovitz did in their KUTATÓ system.

Both approaches are equally valid and, in fact, complement each other. Some hybrid methods [27,28] use a dependency-based method to extract a preliminary structure and then test on orientations by finding those orientations that yield a joint distribution closest to the one induced by the data.

In any case, each method requires the definition of a way of measuring and testing conditional dependence. In probabilistic settings the $\chi^2$ test is used. Other measures have been proposed, nevertheless [1]. However, in other uncertainty calculi, dependence has to be defined in other terms. In our case, in possibility theory, dependence measures are of different classes [11]. We have introduced in the previous sections a graded dependence measure based on the idea of similarity between distributions.

Now, we will comment which measures of the information associated to a possibility distribution are used in possibility theory. When information is formalized by means of a probability distribution, the corresponding information measures is that of *entropy*, which is based on the Shannon measure. In possibility theory comparisons about the informativeness of a given set on which a possibility distribution has been defined are measured in terms of the number of possible alternative values. Informativeness is related to *precision*. The less number of alternatives, the more specific, and hence precise is information about the given set. This kind of measurement is the one that is embodied by the Hartley measure of information [12,19].

Now, an extension of this kind of information measure is applied to possibility distributions defined on a set. This gives as a result the definition of *non-specificity*. A common measure of non-specificity is *U-uncertainty* which is an extension of the Hartley measure of information.

**Definition 4.1** (*U-uncertainty*). Given a variable $X$ with domain $\{x_1 \ldots x_n\}$ and an associated possibility distribution $\pi_x(x_i)$ the $U$-uncertainty for $\pi(x)$ is

$$U(\pi(x)) = \int_0^1 \lg_2 \operatorname{card}(X_\rho)\, \mathrm{d}\rho,$$

where $X_\rho$ is the $\rho$ cut for $X$. That is, $X_\rho = \{x_i \text{ such that } \pi(x_i) \geqslant \rho\}$.

$U$-uncertainty can be extended for joint and conditional distributions in the following way:

**Definition 4.2** (*Joint U-uncertainty*). Given a variable $X_1 \ldots X_n$ variables with associated possibility distributions $\pi_{X_1} \ldots \pi_{X_n}$ their joint non-specificity measured as $U$-uncertainty is:

$$U(\pi_{X_1} \ldots \pi_{X_n})) = \int_0^1 \lg_2 \operatorname{card}(X_{1_\rho} \ldots X_{n_\rho})\, \mathrm{d}\rho$$

**Definition 4.3** (*Conditional U-uncertainty*). Given two variables $X$, $Y$ with associated possibility distributions $\pi_X$, $\pi_Y$ their conditional non-specificity measured as conditional $U$-uncertainty is

$$U(\pi_X(x)|\pi_Y(y)) = \int_0^1 \lg_2 \frac{\operatorname{card}(X_\rho \times Y_\rho)}{\operatorname{card}(Y_\rho)} \mathrm{d}\rho.$$

Note that $U(X|Y) = U(X, Y) - U(Y)$. The $U$-uncertainty of the joint possibility distribution induced by a DAG can be calculated by means of the previous definition. We will do it in two steps.

**Definition 4.4** (*Parent–children non-specificity*). Let $G$ be a DAG representing the conditional independence relationships existing between the variables in a domain $U = \{x_1 \ldots x_n\}$. For any given variable $x_i$ with parent set $pa_i$, the parent–children non-specificity is

$$U(x_i|pa_i) = U(x_i, pa_i) - U(pa_i),$$

when $pa_i = \emptyset$ then $U(x_i|pa_i) = U(x_i)$.

**Definition 4.5**. (*DAG non-specificity*). For a given DAG $G$ defined on the same domain as in the previous case the DAG non-specificity is:

$$U(G) = \sum_{x_i \in U} U(x_i|pa_i).$$

Now it is possible to devise a hybrid algorithm for the recovery of possibilistic networks that makes use both of conditional dependence information discovered by means of the previously defined similarity-based dependency measure and the measure of DAG non-specificity just introduced.

### 4.1. POSSCAUSE

The definition of graded conditional dependence between variables allows for the identification of *groups* or *clusters* of variables that are dependent up to a certain level on a given variable. Note that this dependence can be a *mar-*

*ginal* or *conditional* one. From the structural point of view, a DAG representing a dependence model for a given domain, can be seen as the composition of several subgraphs built up with the direct causes and effects of each variable. Let us call such subgraphs *reduced sheaths*. We borrow the term 'sheath' from Huete [7], although we use it in a slightly different way: by 'sheath' he meant the set both of direct and indirect causes and effects in a *singly directed graph*.

**Definition 4.6** (*Reduced sheath*). For a node $x_i$ in a *DAG* representing the conditional independence relationships existing in a given domain $U$, with a set of marginally dependent variables $\lambda_{x_i}$, the reduced sheath of $x_i$, $\rho_{x_i}$, is the set of those variables $y$ in $\lambda_{x_i}$ such that $y \in Adj(x_i)$ where $Adj(x_i)$ is the set of variables in the DAG that are adjacent to $x_i$.

For any couple of variables $\{y, z\}$ $y, z \in \rho_{x_i}$ the following conditions hold:
1. $I(y|x_i|z)$,
2. $\neg I(x_i|z)$,
3. $\neg I(x_i|y)$.

We will call *focus of a sheath* the variable $x_i$ around which the sheath is built.

Now, any given variable in the domain $x_i$ is either the focus of a reduced sheath or a direct cause or effect of another variable $x_j$ in $U$ such that $I(x_i|\emptyset|y_i)$ or it is an indirect cause of effect of another variable $I(x_i|x_k|x_j)$. That is, it simultaneously belongs to a sheath and it is the focus of its own sheath. Variables with no ancestors are the center of a reduced sheath with no direct causes and members of the sheath of another variable, with respect to which they are direct causes. Variables with no descendants, analogously, are the center of a sheath and/or belong to other variable's sheath as direct effects.

Note that if $I(x_i|x_j|x_k)$, then $x_j$, depending on link orientation, $x_k$ is a direct effect of $x_i$ and a direct cause of $x_j$ or vice versa. Note also that it may be the case that $x_i$ and $x_j$ do not become conditionally dependent until more than one variable is tested. That is, the assertion $I(x_i|x_k|x_j)$ may be false but $I(x_i|x_{k_1} \cup x_{k_2} \ldots x_{k_n}|x_j)$ may be true. In other words, it seems as if one should test for higher order conditional dependencies in order to build the reduced sheath of each variable. This is what makes belief network recovery algorithms so complex. Let us see how this problem of finding high order dependencies between two variables can be circumvented.

Note that if two variables $x_i$ and $x_j$ are dependent conditionally on a third one $x_k$ this last one belongs both to $\rho_{x_i}$ and to $\rho(x_j)$. Variables appearing at each one of the sheaths do obey one simple condition. If $x_k$ belongs to $\rho(x_i)$, then it belongs to $\lambda_{x_i}$, the set of marginally dependent variables of $x_i$ (analogously for $y_i$). Then, in order to build correct sheaths $\rho(x_i)$ and $\rho(x_j)$ it is enough to notice which variables in $\lambda_{x_i}$ are also dependent on $x_j$. Notice that the search for possible candidates for the variables shared by $\rho(x_i)$ and $\rho(x_j)$ is reduced to the ones appearing on $\lambda_{x_i}$ (alternatively $\lambda_{x_j}$), instead of trying to select a variable

from the whole domain, $U$. Notice also, that selection can be guided by the dependence degree between $x_k$ and all the variables outside $\rho(x_i)$ (alternatively $\rho_{x_i}$). Only those variables with the highest dependence degree are selected. This way of understanding the structural relationships in a DAG corresponding to a dependency model allows for an independent building of the different $\rho(x_i)$ for each $x_i \in U$.

Let us see how the corresponding set of direct causes and effects for each variable $x_i$ in $U$ can be found.

Each variable in the domain $x_i$ has two important associated sets.

1. Set $\lambda_{x_i}$.
2. Set $\delta_{x_i}$, the set of variables depending on variables in $\lambda_{x_i}$.

For any variable $x_j$ in $\lambda_{x_i} \neg I(x_i|z)$ and $\neg I(x_i|y)$ hold. For any variable $x_k$ in the set $\delta_{x_i}$ the property $I(x_i|x_j|x_k)$ holds for any variable $x_j$ in $\lambda_{x_i}$.

As we said before, variables in $\rho(x_i)$ are determined from the ones appearing in $\lambda(x_i)$.

If an assertion $I(x_i|x_k|x_j)$ is found, then $x_k$ belongs both to $\rho(x_i)$ and to $\rho(x_j)$. Note that $x_i$ belongs to $\delta(x_j)$.

The construction of $\rho(x_i)$ is made then in the following way.

Process **Build** $\rho(x_i)$

- **Input**:
  - database $D$ on variables $x_1 \ldots x_n$ in $U$
  - $M$, the dependence model extracted from Database $D$, i.e., a list of conditional dependence assertions of the form $I(x|y|z)$ or $I(x|\emptyset|z)$ with $x$, $y$ and $z \in U$
- **Output**: $\rho(x_i)$, the reduced sheath for $x_i$

1. Let $\rho(x_i) = \{\ \}$
2. If $I(x_i|\emptyset|x_j) \in M$) then $\rho(x_i) = \rho(x_i) \cup x_j$
3. If $I(x_i|\emptyset|x_j)$
4. For each $x_k \in U$ such that $I(x_i|x_k|x_j)$ then
   (a) If $x_k$ is not in $\rho(x_i)$ then $\rho(x_i) = \rho(x_i) \cup x_k$
   (b) If $x_k$ is not in $\rho(x_j)$ then $\rho(x_j) = \rho(x_j) \cup x_k$

The cost of the $\rho_{x_i}$ building is in the number of marginal dependence assertions existing in $M$. Each variable $x_i$ in $U$ may be marginally dependent with the rest of variables in the domain, so the cost of this process is $O(n)$, $n$ being the cardinality of $U$.

Note that step 4b is implemented as a message from the process building $\rho_{x_i}$. Details are commented further on.

The result of each process operations are several partial subgraphs representing $\rho(x_i)$ for each $x_i$ in $U$. However, this is only a partial result representing the *skeleton* of the corresponding graph. To be correctly built, each $\rho(x_i)$ must be oriented.

**Orientation of** $\rho(x_i)$

- **Input**: $\rho(x_i)$
- **Output**: the minimum non-specificity oriented subgraph corresponding to $\rho_{x_i}$

1. Let $NS = 0$
2. Let result $= \{\ \}$
3. For each $x_j, x_k$ in $\rho_{x_i}$
4. Find the minimum non-specificity configuration $\min_{pc}$ of the set $\{x_j \rightarrow x_i \rightarrow X_k, y \leftarrow x_i \rightarrow x_k, x_j \leftarrow x_i \leftarrow x_k\}$
5. Let result = result $\cup \min_{pc}$, if it does not create a cycle

Finding the minimum non-specificity parent–children set of $x_i$ is equivalent to testing for each pair of variables $y, z$ in $\rho(x_i)$ which of the three above mentioned orientations reduces in a greater amount the accumulated non-specificity.

The relationship between non-specificity and the degree of dependence between variables allows us to ensure that looking for higher dependency variables results in lower non-specificity graphs, in ch. 4 of [26] there is a full discussion of this issue.

**The POSSCAUSE Algorithm**

- **Input**: A Database $D$ of cases on domain $U$, a threshold, $\alpha$, for similarity tests
- **Output**: A minimum non-specificity DAG, $G$, representing the underlying dependency model on $U$ defined by cases in $D$

1. Let $G = \{\ \}$, the empty graph
2. Build the marginal dependence Model $M$ from $D$ by detecting pairwise dependencies for all $x$, $y$ in $U$
3. For each $x_i$ in $U$
   (a) Build $\rho(x_i)$
   (b) Orient $\rho(x_i)$
4. For each $x_i$ in $U$, let $G = \text{Merge}(G, \rho(x_i))$.
5. Output $G$

Merge$(G, \rho(x_i))$ creates a graph from the previously existing graph G connecting common nodes and avoiding cycles.

### *4.2. Parallel implementation*

Currently, the POSSCAUSE algorithm has been implemented on a IBM-SP2 computer under PVM-E software. It is organized around a supervising process and several local processes. The supervisor process is in charge of global operations: marginal dependency table construction (that is, extracting the

dependency model from data) and reduced sheath fusion. Each subordinate process can perform two operations: reduced set construction and link orientation.

In principle, each variable should be allotted a single process and a single processor, but usually, due to system overload this is not the case. Several policies are being used to obtain the most efficient load distribution among processors.

The supervisor process starts by building the global dependency table or dependency model. The supervisor process spawns several subordinate processes. Each one of them is in charge of a section of the database, which is conceptually divided into sets of the same number of records. Over this part of the domain marginal and conditional dependencies between variables are searched for. This results in the construction of the $\lambda_{x_i}$ set for each variable $x_i$ in the domain and also a list of conditionally dependent variables for each $x_i$ in the domain.

When each process finishes it issues a signal to the supervisor process. Upon completion of all process the next step starts, which is the core of the POSSCAUSE algorithm.

**POSSCAUSE: Parallel version, supervising process**

- **Input**: a list of variables $x_1 \ldots x_n$ in $U$, the corresponding database of cases $D$, an $\alpha$ value
- **Output**: A minimum non-specificity DAG representing the underlying dependency model on $U$ defined by cases in $D$.

1. Build the pairwise dependency table for all $x,y$ in $U$
2. Let $G = G_{\emptyset}$, the empty DAG
3. Let finished = 0
4. Wait until finished = n−1
5. For $i = 1$ to $n$
   (a) send(i,Dep($x_i$))
6. If Receive($\rho(x_i)$) then finished = finished+1
7. For $i = 1$ to $n$, $G$ = Merge($G, \rho$)$x_i$
8. Output $G$

It is important to note that in performance tests, the parallelization of this phase was far superior to its serial counterpart whenever a uniform distribution of the data was present. If this premise was not true, then the efficiency improvement gained by parallelization is not so good.

In fact, the loop that appears here is just notational convention. The true loop of the serial version of POSSCAUSE is converted into a collection of parallel process, each one in charge of the construction of a single causal subgraph.

### 4.3. Programming model

In order to parallelize the POSSCAUSE algorithm the master/slave programming model has been adopted. As we will see, some parts of the algorithm, specially those that are needed for ancillary calculations, should be done in sequential fashion while others pose no problems for parallelization.

There is a need for planning the execution of tasks in the most efficient way so as to optimize the use of physical processors by the different tasks. The master process will be in charge of this planification tasks and will distribute the workload among all tasks.

### 4.4. Algorithm structure

As we mentioned before, each process should take care of the construction of a single causal subgraph. However, given that the number of available processors cannot be, in general, equal to the maximum number of variables in a DAG, there is a hidden distribution of tasks among processes on several physical processors.

Theoretically, the master process should only deal with initiation of the learning algorithm, the coordination of the different parts and the final fusion process. This implies that the master process should have an idea of the volume of tasks that each other process needs. Using this information, it should provide to the process in charge of each variable the data that it needs for constructing the appropriate causal subgraph: dependence degree, similarity degree, a list of variables and the corresponding possibility distributions. Also, in detecting that a variable must 'migrate' to the causal subgraph of another variable it has to direct the traffic of variable sets between processes.

However, as we will see, the parameters needed for planning in a statical way vary with the number of variables in a causal graph. So, planification has to be done *dynamically* comparing the available resources (free processes) and the tasks that are to be done.

There are some tasks that should be done sequentially, as the merging of the causal subgraphs, that cannot be done after each of the subgraphs is constructed.

The main tasks are, the, same as the algorithm steps: possibility distributions calculation, dependence assertion calculations, causal subgraph construction (which is further divided into dependency set calculation and true causal subgraph construction), orientation of causal subgraph and final fusion.

## 5. Problem partition

The whole problem can be broken into different subproblems that correspond to the different tasks. Let us look at each one in turn.

1. *Building the set of marginal dependent variables*: For each variable $X_i$, the variables that are marginally dependent on it are to be found. Note that there is no need to coordinate the different processes since, although there will exist some level of redundancy due to symmetry, each variable can be considered at this step independently of the other ones.

2. *Building the causal subgraph*: For each variable $X$ a first initial subgraph can be from the set of marginally dependent variables. Note that this means finding among those variables in the $\delta(x)$ set those that $I(X|Y|Z)$ in order to migrate variable $Z$ outside the causal subgraph of $X$.

A set of interacting variables results from this step as we have commented before. Clearly, this list is not complete until all initial subgraphs have been built. So, this will be a task for the master process.

3. *Orientation of causal subgraphs*: Again this can be done in any order although this is calculation intensive step because it requires the calculation of non-specificity.

4. *Subgraph fusion*: Subgraph fusion cannot be done until all graphs are oriented, so this is clearly a task to be performed sequentially.

We will describe each process in turn. However, previously to this we have to comment on the planning process with more detail.

### 5.1. Possibility distribution calculation

The calculation of the needed possibility distributions requires exploring the database in order to construct the possibility distributions for the different variables. Marginal possibility distributions can be calculated as if the variables were independent. So, a process is generated for each variable. The whole set of cases in the database can be partitioned into equal fragments. Each one contains a subset of successive cases in the database. The size of each fragment is calculated so as to reduce to the minimum the communication between slave and master processes.

If the database contains $M$ cases and the optimal size for communication is $P$, each fragment is $M/P$ cases long. In this way, a certain improvement in the cost of the calculation of the possibility distributions is attained by parallelizing an otherwise sequential process.

An approximated probability distribution is calculated in this fashion and then it is transformed into a possibility distribution applying the maximum normalization.

Note that the construction of the probability distribution requires counting how many different instances for the different instantiation of the variables exist in the database.

The master process issues $P$ starting signals (one for each available processor) and has to merge the $M/P$ partial distributions. For each of them the max-

imum value of probability is also recovered. The corresponding normalization is done for each variable. This gives marginal distributions.

For each instance of the different combination of values, a similar transformation is performed.

For each variable in the domain a marginal dependence test is done using the joint and marginal possibility distributions previously recuperated. This is done in order to obtain $\delta(X_i)$.

In principle, this should be a straightforward step. Each slave task receives the order to issue dependence tests on $n - 1$ variables. However, due to the fact that dependence requires more calculations for variables with greater cardinality, those tasks that are assigned the construction of variables with less possible values will end earlier. This circumstance makes the static planification of tasks too complex and not convenient. Dynamic planification is advisable.

A *pool of tasks* solution is adopted. Each process can have $K$ causal graphs to calculate. Constant $K$ is a function of the *chunksize* of the system. That is, of the optimum volume of data for communication. With the pool system, more costly subgraphs are the first ones to be allocated. The pool is always kept in ordered fashion, from more to less costly processes. Cost is calculated in terms of the number of variables for each subgraph and their cardinality.

Whenever a task finishes its work, the master process looks for a variable $X_i$*et* whose causal subgraph has not been built *yet*. Then a job is created and sent to the pool. Each job uses as data a variable, its list of dependencies as well as the conditional dependency assertions involving just the related variables.

Each one of the processes receives a subset of the total database corresponding to its variable.

If $T$ is the total number of occurrences in the database, and $N$ is the number of processors, each processor is allocated the job of counting $T/N$. For any size of the database all available processors are used.

Variables which are marginally dependent on a certain level $\gamma$ with the variable that is the focus of the subgraph are included in this set.

Order in the pool of tasks is given by the number of possible values a variable can take.

Each process needs the possibility distribution corresponding to the variables. When a process ends, if there are some subgraphs to be calculated, it is assigned a new set of variables. See [26] for a full discussion of the different process scheduling policies.

## 6. Experimental results

As we mentioned before, POSSCAUSE has been implemented on a SP2 computer under PVM-E. There are several questions that can be tuned when dealing with such an architecture. First of all, given the division work among

master and slaves, it is important to decide on the *chunksize* in order to find the best configuration. We have run tests on the ALARM database in order to experiment with several chunksizes and number of processors. These parameters are used to assess the degree of speedup and efficiency that is obtained in POSSCAUSE.

### 6.1. The ALARM Database

The Alarm database [4,16,6] contains 20,000 cases from the anesthetic emergency domain. The DAG that corresponds to the knowledge about the domain that is shown in Fig. 1. It was generated by Cooper et al. [6] by the Monte Carlo simulation algorithm for belief networks developed by Henrion [14]. In our tests we used two partial datasets of 5000 and 10,000 cases as well as the whole ALARM database.

The resulting DAG for 5000 is shown in Fig. 2. As it can be seen, some links are missing and others are incorrectly oriented. The proportion of erroneous links is similar to other belief network learning algorithms as K2 [6] and CB [28]. Let us remark, however, that POSSCAUSE is more robust to the number of data. That is, differences between DAGs recovered from 5000 and 10,000 data or from 10,000 and 20,000 are not so high as in the corresponding experimental results of the above mentioned algorithms. This may indicate that possibilistic methods are more robust to data variations which is in accordance to a framework that deals with imprecision.

A thorough analysis of this behavior can be consulted in [26] in particular a comparison between the results of POSSCAUSE against K2 [6] and CB [28].

There are other things to note from this result. The separation in two different DAGs may be due to low evidence on the dependence between variable 27 and variable 11. We are now making other tests with several $\alpha$ values and also

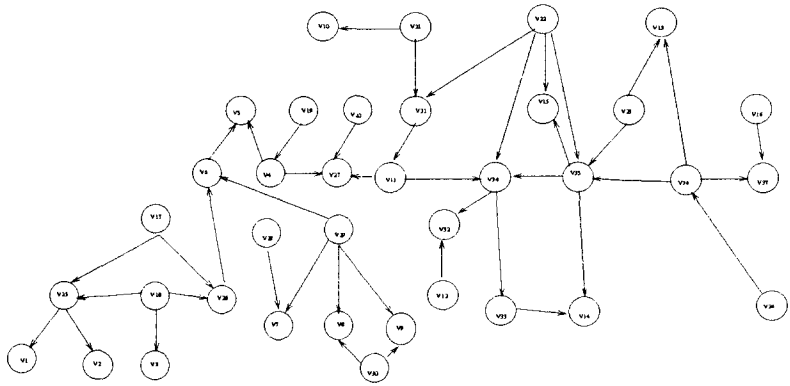

Fig. 1. The original ALARM DAG.

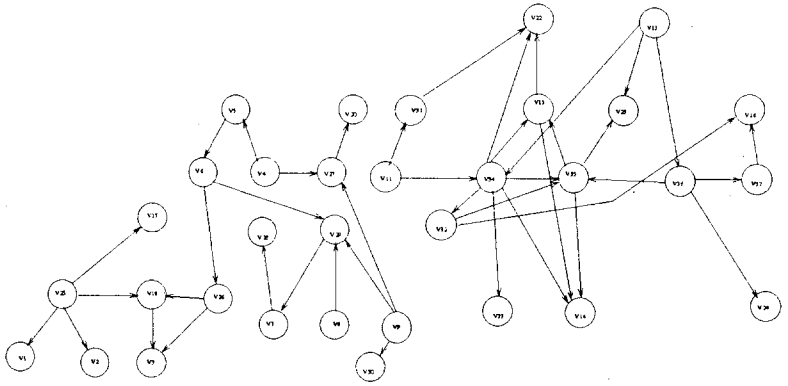

Fig. 2. The resulting DAG after applying POSSCAUSE on 5000 cases.

with different data sets. Cooper and Herskovtitz in their experiments, although done on 10,000 cases give no clue as which cases are these. May be a random selection of cases could approximate better the connection between the known disconnected graphs.

As to the importance of the configuration of the process model, there were several tests made measuring speedup and time efficiency by using different numbers of processors and chunksizes. Figs. 3–5 show the different behaviors. As it can be seen in Fig. 3, maximum speedup is obtained with a chunksize of 2 and using 4 processors. It is natural to see that with six processors the communication overhead needed imbalances the gain obtained by using more processors.

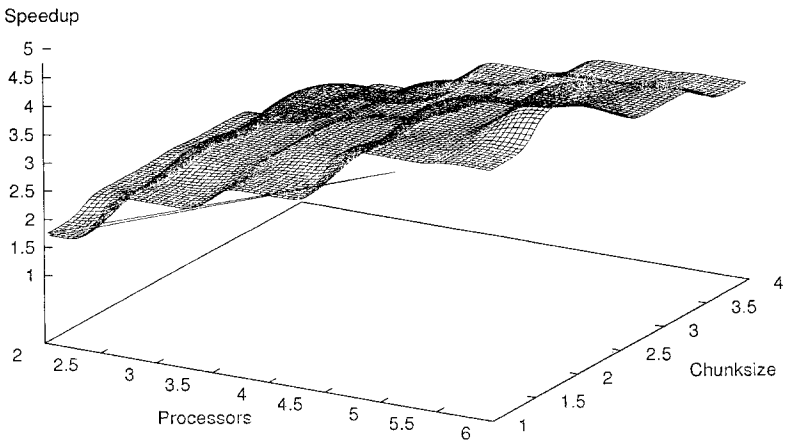


Fig. 3. Speedup in relationship to number of processors and chunksize.

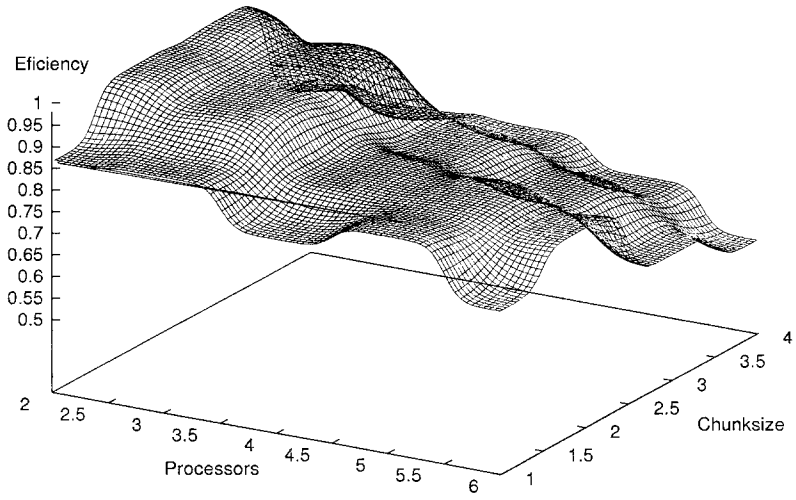


Fig. 4. Efficiency versus number of processes and chunksize.

Time exhibits the expected behavior in relationship to the increase of the number of processors. As more processors are used, the time involved in doing the calculations decreases. This is generally true but when the number of processors is higher, there appears a slight decrease in efficiency due to the raise of

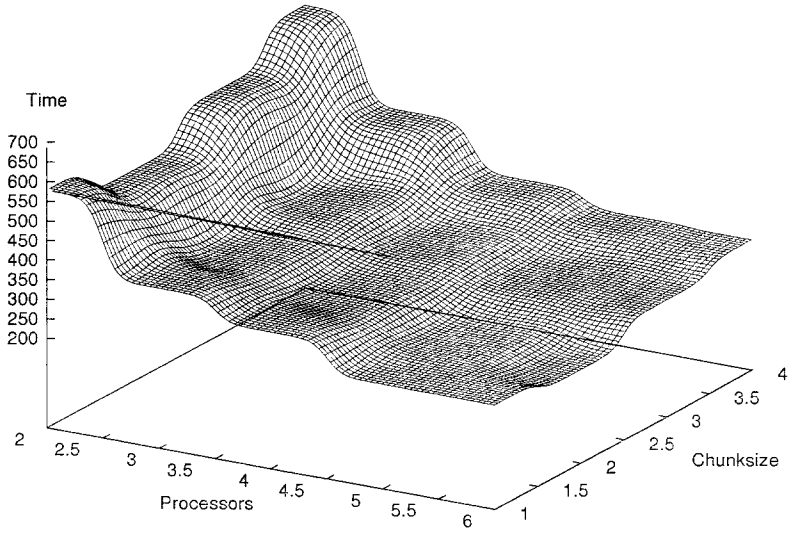


Fig. 5. Time versus number of processes and chunksize.

communication operations among processors. This happens whatever the chunk size is. Fig. 4 shows that best results in efficiency are found when using from two to four processors with a chunk size of 2.

The fact that system uses distributed memory, induces a loss of efficiency due to communication costs and slow working of the network.

Monitoring of execution showed that most of the processing time is absorbed by the preparation phase. That is the process that built the probability and possibility tables. For this reason we ported all the PVM calls onto a shared memory system. Previous tests on the recovery of partial graphs seem to indicate an increase of efficiency by the reduction of the previously mentioned communication costs.

## 7. Discussion and future work

The parallel version of the POSSCAUSE algorithm has been presented. The algorithm is based on the creation of clusters of highly dependent variables. The construction procedure ensures that the recovered DAG corresponds to a joint possibility distribution that minimizes non-specificity, that is, it recovers the most precise DAG given the available data.

Future developments include studying the relationship between precision and number of data available. This will allow a more directed selection of $\alpha$ values according to the quality of the data. Also, the integration of previously known causal information is being studied in order to use it to guide selection of possible DAG structures. The integration of previous knowledge from experts and this raises the question of which is the information that has to be given a greater confidence: the one coming from the experts in the form of a priori knowledge or the one extracted from the database. Methods for revising the a priori theory in possibilistic settings are under study. The goal is to be able to revise the causal theory of the domain as it is being built in a dynamic way according to the knowledge being gathered and its relationship to previous knowledge on causal relationships.

As to the efficiency problems shown in the parallelization process, the first steps to port the algorithm onto a shared memory architecture indicate a source of efficiency gain. However, further study is required to devise more efficient ways of calculating conditional distribution tables.

## References

[1] S. Acid, L.M. De Campos, A. González, R. Molina, N.P. de la Blanca, Learning with castle, in: R. Kruse, P. Siegel (Eds.), Symbolic and Quantitative Approaches to Uncertainty, Lecture Notes in Computer Science, no. 548, Springer, Berlin, 1991.

[2] S. Acid, L.M. De Campos, Approximations of causal networks by polytrees: an empirical study. in: B. Bouchon-Meunier, R.R. Yager, L. Zadeh (Eds.), Advances in Intelligent Computing, Lecture Notes in Computer Science, no. 945, Springer, Berlin, 1995, pp. 149–158.
[3] J. Béjar, U. Cortés, Linneo: herramienta para la adquisición de conocimiento y generación de reglas de clasificación en dominios poco estructurados, in: Actas del III Congreso Iberoamericano de Inteligencia Artificial (IBERAMIA92), 1992.
[4] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, F.G. Cooper, The Alarm monitoring system: A case study with two probabilistic inference techniques for belief networks, in: Proceedings of the second European Conference on Artificial Intelligence in Medicine, London, 1989, pp. 247–256.
[5] C.K. Chow, C.N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory 14 (1968) 462–467.
[6] G. Cooper, E. Herskovitz, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (1992) 320–347.
[7] L.M. De Campos, J.F. Huete, Learning non-probabilistic belief networks, in: Proceedings of the second European Conference on Quantitative and Symbolic Approaches to Reasoning under Uncertainty, 1993.
[8] L.M. De Campos, Independence relationships in possibility theory and their application to learning belief networks, in: G. Della Riccia, R. Kruse, R. Viertl (Eds.), Mathematical and Statistical Methods in Artificial Intelligence, CISM Courses and Lectures, no. 363, Springer, Berlin, 1995, pp. 119–130.
[9] M.J. Drudzel, H.A. Simon, Causality in Bayesian belief, in: Proceedings of the Ninth Conference on Uncertainty in Artifical Intelligence, Morgan Kaufmann, San Mateo, CA, 1993, pp. 3–11.
[10] P. Fonck, Propagating uncertainty in directed acyclic graphs, in: Proceedings of the fourth IPMU Conference, Mallorca, 1992.
[11] P. Fonck, Reseaux d'inference pour le raisonnement possibiliste, Ph.D. Thesis, Université de Liege, 1993.
[12] R.V.L. Hartley, Transmission of information, The Bell Systems Technical Journal 7 (1928) 535–563.
[13] D.A. Heckerman, A Bayesian approach to learning causal networks, Technical Report MSR-TR-95-04, Microsoft Research Advanced Technology Division; also in Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 1995, pp. 285–295.
[14] M. Henrion, Propagating uncertainy in Bayesian networks by logic sampling, in: J.F. Lemmer, L.N. Kanal, Uncertainty in Artificial Intelligence, vol. 2, North-Holland, Amsterdam, 1988.
[15] E.H. Herskovitz, G. Cooper, Kutató: an entropy-driven system for the construction of probabilistic expert systems from data, in: Proceedings of the sixth conference on Uncertainty in Artificial Intelligence, 1990.
[16] E.H. Herskovitz, Computer-based probabilistic networks construction, Medical Information Sciences Section, Stanford University, 1991.
[17] J.F. Huete, L.M. De Campos, Learning causal polytrees, in: R. Kruse, M. Clarke (Eds.), Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science, no. 747, Springer, Berlin, 1993.
[18] J.F. Huete, Aprendizaje de redes de creencia mediante la detección de independencias: modelos no probabísticos, Ph.D. Thesis, Universidad de Granada, Granada, Spain, 1995.
[19] G. Klir, T. Folger, Fuzzy Sets Uncertainty and Information, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[20] S. Kullback, R.A. Leibler, Information and sufficiency, Annals of Mathematical Statistics 22 (1951) 79–86.
[21] W. Lam, F. Bacchus, Learning Belief Networks an approach based on the MDL principle, Computational Intelligence (1994) 104–127.

[22] J. Pearl, A. Paz, Graphoids: a graph-based logic for reasoning about relevance relations, Technical Report, Cognitive Science Laboratory, Computer Science Department, University of California, Los Angeles, 1985.
[23] J. Pearl, T. Verma, A theory of inferred causation, in: Proceedings of the Second International Conference on Knowledge Representation and reasoning, Morgan Kaufmann, San Mateo, CA, 1991.
[24] T. Rebane, J. Pearl, The recovery of causal poly-trees from statistical data, in: L.N. Kanal, T.S. Levitt, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, North-Holland, Amsterdam, vol. 3, 1989.
[25] R. Sangüesa, U. Cortés, Learning causal networks from data: a survey and a new algorithm to learn possibilistic causal networks from data, AI Communications 10 (1) (1997) 31–62.
[26] R. Sangüesa, Learning possibilistic networks from data, Ph.D. Thesis, Technical University of Catalonia, Barcelona, Spain, 1997, http://www.lsi.upc.es/~sanguesa.
[27] M. Singh, M. Valtorta, An algorithm for the construction of Bayesian network structures from data, in: Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1993, pp. 259–265.
[28] M. Singh, M. Valtorta, Construction of Bayesian network structures from data: a survey and an efficient algorithm, International Journal of Approximate Reasoning 12 (1995) 111–131.
[29] T. Verma, Causal networks: semantics and expressiveness, in: R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer, Uncertainty in Artificial Intelligence, vol. 4, Elsevier, Amsterdam, 1989.
[30] T. Verma, J. Pearl, An algorithm for deciding if a set of observed independencies has a causal explanation, in: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Los Altos, 1992, pp. 323–330.